October 2005

Discrete Abstractions for Robot Motion Planning and Control in Polygonal Environments

Calin Belta
Boston University

Volkan Isler
University of California, Berkeley

George J. Pappas
University of Pennsylvania, pappasg@seas.upenn.edu

Copyright 2005 IEEE. Reprinted from *IEEE Transactions on Robotics*, Volume 21, Issue 5, October 2005, pages 864-874.

Discrete Abstractions for Robot Motion Planning and Control in Polygonal Environments

Abstract
In this paper, we present a computational framework for automatic generation of provably correct control laws for planar robots in polygonal environments. Using polygon triangulation and discrete abstractions, we map continuous motion planning and control problems, specified in terms of triangles, to computationally inexpensive problems on finite-state-transition systems. In this framework, discrete planning algorithms in complex environments can be seamlessly linked to automatic generation of feedback control laws for robots with underactuation constraints and control bounds. We focus on fully actuated kinematic robots with velocity bounds and (underactuated) unicycles with forward and turning speed bounds.

Keywords
Bisimulation, control, discrete abstraction, hybrid system (HS), motion planning, triangulation

Discrete Abstractions for Robot Motion Planning and Control in Polygonal Environments

Calin Belta, *Member, IEEE*, Volkan Isler, *Member, IEEE*, and George J. Pappas, *Senior Member, IEEE*

***Abstract*—In this paper, we present a computational framework for automatic generation of provably correct control laws for planar robots in polygonal environments. Using polygon triangulation and discrete abstractions, we map continuous motion planning and control problems, specified in terms of triangles, to computationally inexpensive problems on finite-state-transition systems. In this framework, discrete planning algorithms in complex environments can be seamlessly linked to automatic generation of feedback control laws for robots with underactuation constraints and control bounds. We focus on fully actuated kinematic robots with velocity bounds and (underactuated) unicycles with forward and turning speed bounds.**

***Index Terms*—Bisimulation, control, discrete abstraction, hybrid system (HS), motion planning, triangulation.**

I. Introduction

MOTION planning and control of robots in complex environments is a fundamental problem that has received a lot of attention. The vast literature on this topic can be roughly divided into two schools of thought. The first focuses on the complexity of the environment while assuming that the robot is fully actuated with no control bounds, and includes approaches based on roadmap methods such as Voronoi diagrams, visibility graphs, and freeway methods [23], potential fields [19], [23], navigation functions [20], cellular decompositions, probabilistic roadmaps [24], and hierarchical path [11] and task [12] planning. The other school of thought focuses on the detailed dynamics of the robot, assuming simple environments. Some of these approaches are differential geometric [21], some exploit concepts such as flatness [30], while others use input parameterizations [26], [34], or discontinuous control laws [3].

There has been interest recently in creating computational frameworks combining the discrete algorithms capturing the complexity of the environment with the continuous approaches modeling the kinematics or dynamics of the robots. For example, in [33], Delaunay triangulations are used to discretize the environment, and cubic spline representations are employed to meet robot kinematic constraints. Generation of smooth curves with curvature guarantees is linked to a probabilistic roadmap planner in [22], while a sampling-based roadmap method that deals with nonholonomic constraints was developed in [8]. Polygonal partitions of planar environments followed by assignment of vector fields obtained as solutions of Laplace's equation in each of the regions were considered in [9].

In this paper, we bring ideas from formal analysis of hybrid systems (HSs) to build a framework in which solutions of discrete algorithms dealing with the complexity of the environment automatically generate provably correct robot-control laws. We first focus on point-like planar fully actuated robots with control bounds moving in polygonal environments, and then present an extension to unicycles with control bounds and negligible size. Task specifications given in terms of regions to be reached or avoided by the robot naturally induce a partition of the polygonal environment. In this paper, as in [31], we focus on triangulations, and use the quotients produced by such partitions (finite graphs) as a discrete framework for formulating and solving motion-planning problems. A motion plan is, therefore, a path in the graph that can be either explicitly specified, or can be determined as a solution of an optimal problem on the graph or of a temporal logic specification [32]. Other examples include solutions to discrete games. For example, in the visibility-based game presented in [15] and [16], the winning strategy of the pursuer is to randomly generate paths on this graph. The focus of this paper is not on producing a discrete solution, but rather on generating robot-control laws implementing such discrete specifications. To this goal, we construct two types of vector fields (feedback controllers) for each triangle: (I) vector fields driving all initial states in a triangle through an arbitrary facet in finite time, and (II) vector fields making the triangle an invariant. The resulting HS and the partition quotient satisfy an equivalence relation called *bisimilarity* [2], which is the general framework of equivalence between transition systems.

Motivated by [5] and [13], the focus of this paper is on affine vector fields, which are uniquely determined by their values at the vertices of a full-dimensional simplex in an arbitrarily dimensional Euclidean space, and whose restrictions to such simplices are convex combinations of the values at the vertices. Based on this property, and following the same lines of [13], we first derive necessary and sufficient conditions for the existence of controllers of type (I). We then derive equivalent conditions

Manuscript received September 23, 2004; revised February 16, 2005. This paper was recommended for publication by Associate Editor L. Parker and Editor S. Hutchinson upon evaluation of the reviewers' comments. The work of C. Belta was supported in part by the National Science Foundation under NSF CAREER 0447721 and under NSF CNS 0410514. The work of G. Pappas was supported in part by the National Science Foundation under NSF CAREER 0132716 and under NSF ITR 0324977, and in part by the Army Research Office under MURI DAAD 19-02-01-0383.

C. Belta was with the Department of Mechanical Engineering and Mechanics, Drexel University, Philadelphia, PA 19104 USA. He is now with the Department of Manufacturing Engineering, Boston University, Boston, MA 02446 USA (e-mail: cbelta@bu.edu).

V. Isler is with the Center for Information Technology Research in the Interest of Society (CITRIS), University of California, Berkeley, CA 94720 USA (e-mail: isler@eecs.berkeley.edu).

G. J. Pappas is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: pappasg@ee.upenn.edu).

Digital Object Identifier 10.1109/TRO.2005.851359

for the existence of controllers of type (II), and conditions on the robot-control constraints for which any discrete specification can be executed. Finally, for a given discrete path, we give an algorithm for automatic construction of robot-control laws, which are as "smooth as possible."

Compared with the papers combining discrete with continuous path planning mentioned above, our approach has the advantage that it automatically generates provably correct robot-control laws, rather than a path that could be followed by the robot. In contrast with methods based on compositions of behaviors [6], in which the robot controls are derived as gradients of potentials, our method has the advantage of generating controls with guaranteed bounds. Among all the literature on robot planning and control, the work that is most closely related to this paper is [9], where the authors consider a polygonal partition of a planar configuration space, and assign vector fields in each polygon so that initial states in each polygon can only flow to a neighbor through the corresponding common facet. The vector fields are defined as gradients of scalar functions determined as solutions to Laplace's equation. Even though both approaches are motivated by the need for a framework for automatic robot deployment, the method in this paper seems suited for composition and computation. If the robot-control constraints are polyhedral, all the calculations consist of operations on polyhedral sets only, which is much cheaper than solving Laplace's equation. Moreover, given the vertices of a polygon, the triangulation and generation of provably correct feedback controllers implementing a high-level discrete strategy is fully automated in our framework.

The paper is structured as follows. In Section II, we formulate the problem, give the necessary definitions, and present our approach. The main results and the algorithms for automatic generation of vector fields mapping to discrete specifications are given in Sections III and IV. These results are then used in Section V to generate provably correct feedback-control laws for fully actuated kinematic robots and unicycles. Simulation results are shown in Section VI. The paper ends with concluding remarks and a brief exposition of future research directions in Section VII.

II. Problem Formulation and Approach

We consider a fully actuated point like planar robot described by a control system of the form (see *Remark 2* for extension to unicycles)

$$\dot{x} = u, \quad x \in \mathcal{P}, \quad u \in U \tag{1}$$

where x is the position of the robot in a given world frame restricted to a polygon $\mathcal{P}$, which does not change in time, i.e., the environment is static. This polygon can be complex, with a large number of vertices, and it can contain polygonal holes modeling obstacles or undesired regions in the environment. The set U captures the control bounds, and it is assumed to be a convex subset of $\mathbb{R}^2$.

As is usually the case in practice when dealing with complex environments, we assume that the motion-planning task is "qualitatively" specified. Specifically, we are not interested in the exact position x of the robot, but rather in deciding its inclusion in certain regions of interest. For example, in obstacle avoidance, we only need to make sure that the robot does not collide with obstacles of given geometry. Or, to win a visibility-based game, such as the one formulated in [15] and [16], the pursuer only needs to make sure that it is in the same triangle as the evader. Throughout this paper, we assume that these regions are triangles or unions of adjacent triangles. There are several supporting arguments for our choice. First, the problem of triangulating a polygon is well studied, and computationally efficient algorithms are available [28]. Second, as we will see later in the paper, triangles have special properties that can be exploited to map such qualitatively described tasks to discrete transition systems over a finite set of symbols, with automatic generation of provably correct robot-control laws.

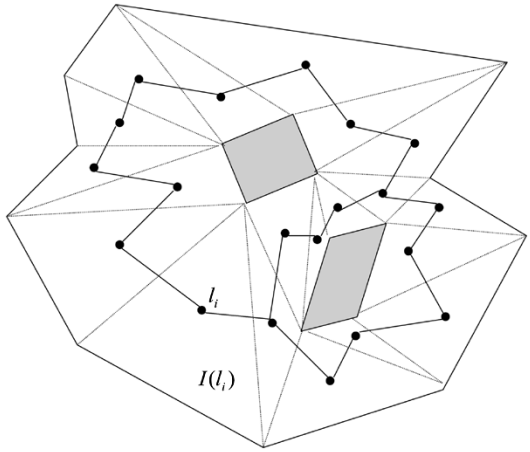

Fig. 1. Triangulation of a planar polygon and the DG.

We label each triangle using a finite set of symbols $L = \{l_1, l_2, \ldots, l_M\}$, and use the notation $I(l_i) \subset \mathcal{P}$ to denote the region of $\mathcal{P}$ contained by triangle l_i, including its boundary. Clearly

$$\mathcal{P} = \bigcup_{l_i \in L} I(l_i). \tag{2}$$

This idea is illustrated in Fig. 1, where the shaded polygons are forbidden regions in a task specification (e.g., obstacles), and the triangulation is achieved by a maximal set of nonintersecting diagonals [28].

Definition 1 (Dual Graph): The *dual graph* (DG) of a triangulation is a graph

$$\mathrm{DG} = (L, t) \tag{3}$$

whose nodes $L = \{l_1, l_2, \ldots, l_M\}$ correspond to the symbols used for labeling the triangles, and the edge set $t \subset L \times L$ denotes a symmetric adjacency relation between the corresponding triangles.

The DG defined by (3) serves as our discrete modeling abstraction for algorithmic motion planning, with specifications given in terms of sets. Its nodes can be seen as "qualitative" robot states, while its edges model state transitions. More formally, the task specifications are given in the language of the DG.

Definition 2 (Language of Dual Graph): The language $\mathcal{L}(\mathrm{DG})$ of DG is the set of all strings $(l_{i_1}, l_{i_2}, \ldots, l_{i_m}), l_{i_j} \in L, i_j \in \{1, \ldots, M\}, j = 1, \ldots, m$, with $(l_{i_j}, l_{i_{j+1}}) \in t, j = 1, \ldots, m-1$.

The high-level specifications given in terms of strings in the language of the DG are determined at a higher hierarchical level, which is beyond the scope of this paper. The focus of this paper is not on determining such strings in the language $\mathcal{L}(\mathrm{DG})$, but rather on creating a computationally efficient and provably correct framework in which a given string is automatically translated to robot-control laws. More formally, we provide a solution to the following problem.

Problem 1: Construct a set $\mathcal{U}$ of state feedback controllers so that, for any string $(l_{i_1}, l_{i_2}, \ldots, l_{i_m}) \in \mathcal{L}(\mathrm{DG})$, there exists $u \in \mathcal{U}$ driving the robot (1) from any initial state $x_0 \in I(l_{i_1})$ through the regions $I(l_{i_1}), I(l_{i_2}), \ldots, I(l_{i_m})$ in finite time, and stays in $I(l_{i_m})$ for all future times.

In other words, if a solution to *Problem 1* exists, then the robot can automatically achieve any discrete specification in the language $\mathcal{L}(\mathrm{DG})$. The set $\mathcal{U}$ will contain two types of controllers: (I) feedback controllers driving the robot from any initial state $x_0 \in I(l)$ to $I(l')$ in finite time, for any $l' \in L$ with $(l, l') \in t$; and (II) feedback controllers driving the robot so that it stays in $I(l)$ for all times, for all initial states $x_0 \in I(l)$, and for all $l \in L$. Indeed, it is easy to see any string $(l_{i_1}, l_{i_2}, \ldots, l_{i_m})$ can be implemented by using controllers of type (I) for $(l_{i_1}, l_{i_2}), (l_{i_2}, l_{i_3}), \ldots, (l_{i_{m-1}}, l_{i_m})$, and a controller of type (II) for l_{i_m}. On the other hand, we need controllers of types (I) and (II) to implement all strings of length two and one, respectively.

We provide a solution to *Problem 1* by constructing vector fields in the polygon $\mathcal{P}$. We construct a set of (maximum) four vector fields for each triangle: one that makes the triangle an invariant, which will lead to a controller of type (II), and (maximum) three that drive all initial states in the triangle to each of its neighbors, which will lead to controllers of type (I). The natural framework for representing such a construction is that of *hybrid systems*, and is presented below. A more general definition of a HS can be found in [1].

Definition 3 (Hybrid System): An HS is a tuple

$$\mathrm{HS} = (\mathcal{P}, \mathcal{Q}, \mathrm{Inv}, f, T, O) \tag{4}$$

where

— $\mathcal{P}$ is its (polygonal) continuous state space (2). $x \in \mathcal{P}$ is called continuous state.

— $\mathcal{Q}$ is its finite set of locations defined by

$$\mathcal{Q} = \{q_{ij} |\ i, j = 1, \ldots, M, \text{ and } i = j \text{ or } (l_i, l_j) \in t\}. \tag{5}$$

$q_{ij} \in \mathcal{Q}$ are called discrete states, or locations. The overall state of the system is, therefore, $(q_{ij}, x) \in \mathcal{Q} \times \mathcal{P}$.

— $\mathrm{Inv}: \mathcal{Q} \to 2^{\mathcal{P}}$ is a map which assigns to each discrete state $q_{ij} \in \mathcal{Q}$ an invariant set defined by

$$\mathrm{Inv}(q_{ij}) = I(l_i). \tag{6}$$

— $f: \mathcal{Q} \to (\mathcal{P} \to T\mathcal{P})$ is a mapping that specifies the continuous flow (vector field) in each location q_{ij}. $f_{q_{ii}}$ keeps the system in the triangle $I(l_i)$ for all times. $f_{q_{ij}}$, with i, j so that $(l_i, l_j) \in t$, drives all initial continuous states $x \in I(l_i)$ to $I(l_j)$ in finite time through the common boundary $I(l_i) \bigcap I(l_j)$.

— $O: \mathcal{Q} \times \mathcal{P} \to L$ is an output map defined as

$$O(q_{ij}, x) = l_i, \quad q_{ij} \in \mathcal{Q}, \quad x \in \mathcal{P}. \tag{7}$$

Note that the number of discrete states (locations) of the HS defined above is $|\mathcal{Q}| = |L| + |t|$, where $|\cdot|$ denotes the cardinality of a set. According to the above definition, while in location $q_{ij} \in \mathcal{Q}$, the system evolves according to

$$\dot{x} = f_{q_{ij}}(x), \quad x \in \mathrm{Inv}(q_{ij}) \tag{8}$$

and outputs l_i. Similar to DG, the language of HS is defined as the set of discrete states (triangles) reached by the system.

Definition 4 (Language of Hybrid System): The language $\mathcal{L}(\mathrm{HS})$ of HS is the set of all strings produced by the output map O as HS evolves in time.

It is easy to see that an HS satisfying *Definition 3* produces the same language as DG, i.e., they are *language-equivalent*.

Remark 1: HS and DG defined above satisfy a stronger notion of equivalence, called *bisimilarity* [14], [29], which implies language equivalence. In these papers, a continuous system or HS is iteratively partitioned until it becomes equivalent with its discrete quotient induced by the partition with respect to reachability properties. In this paper, motivated by robotic motion planning, we consider the inverse problem. Given a set of discrete states and allowed transitions in the form of a DG, we construct a bisimilar HS. Such a system can match any string and provides a framework in which such strings can be composed, as necessary, for example, in the implementation of a discrete game [16].

The main challenge in constructing a solution to our problem is designing vector fields satisfying *Definition 3* of HS. In this paper, we restrict our attention to affine vector fields bounded in convex sets

$$f_{q_{ij}}(x) = A_{q_{ij}} x + b_{q_{ij}} \in U, \quad x \in \mathrm{Inv}(q_{ij}), \quad q_{ij} \in \mathcal{Q} \tag{9}$$

where $A_{q_{ij}} \in \mathbb{R}^{2\times 2}, b_{q_{ij}} \in \mathbb{R}^2$, and $U \subseteq \mathbb{R}^2$ is an arbitrary convex set. For this class of systems, which we call *triangular affine HSs*, we show in Section III that there is a simple and computationally efficient method for characterization of existence and explicit construction of the desired HS. If requirements such as smoothness of the produced control laws over several triangles or minimization of time spent traversing a set of triangles are required, then the algorithm is refined to produce a corresponding solution satisfying the additional requirements in Section IV.

Remark 2: *Problem 1* can be extended to underactuated systems such as unicycles with control bounds and negligible size, by first solving a feedback-linearization problem for a wisely chosen point x on the robot, and by using the above procedure for this point (see Section V). Also, with a bit of extra work and conservatism, fully actuated robots or unicycles with considerable size can be accommodated in this framework. Intuitively, the procedure would consists of three steps: 1) include the robot in a nonrotating polytope by applying all possible rotations around the reference point x; 2) reduce the polytope to the point x by shrinking the environment and enlarging the obstacles; and 3) solve the problem as described in this section for x.

III. Triangular Affine Hybrid Systems

In this section, we provide necessary and sufficient conditions for the existence of a solution to *Problem 1* in the case when the feedback controllers are restricted to be affine. This is equivalent with the existence of an HS (4) that is language-equivalent with the DG (3). To this goal, we characterize all affine vector fields driving all initial states in a triangle through a facet in finite time, or keeping all initial states in a triangle forever. We also provide simple formulas for the construction of such vector fields, which leads to the construction of the triangular affine HS (4), (9). The results are presented for arbitrarily dimensional simplices. To construct affine vector fields in triangles, we use the two-dimensional (2-D) case, when the simplex is a triangle. To achieve smoothness of trajectories as described in Section IV, we use the 1-D case, when the triangle becomes a line segment.

Some of the results in this section are restatements from [5] and [13]. Specifically, *Lemma 1* is an adaptation from [13], which is more suited for computation. A version of *Proposition 2* for control systems with affine drift and constant control directions can be found in [13], and *Proposition 3* is also stated in [5].

A. Affine Functions in Simplices

This section presents an interesting property of an affine function defined in a simplex. It is uniquely determined by its values at the vertices of the simplex, and its restriction to the simplex is a convex combination of these values.

Let $N \in \mathbb{N}$, and consider $N+1$ affinely independent points $v_1, \ldots, v_{N+1}$ in the Euclidean space $\mathbb{R}^N$, i.e., there exists no hyperplane of $\mathbb{R}^N$ containing $v_1, \ldots, v_{N+1}$. Then the simplex S_N with vertices $v_1, \ldots, v_{N+1}$ is defined as the convex hull of $v_1, \ldots, v_{N+1}$

$$S_N = \left\{ x \in \mathbb{R}^N | x = \sum_{i=1}^{N+1} \lambda_i v_i, \sum_{i=1}^{N+1} \lambda_i = 1, \lambda_i \geq 0 \right\}. \quad (10)$$

For $i \in \{1, \ldots, N+1\}$, the convex hull of $\{v_1, \ldots, v_{N+1}\} \setminus \{v_i\}$ is a facet of S_N and is denoted by F_i. Let n_i denote the corresponding unit outer normal vector.

For $r \in \mathbb{N}$, let $f : \mathbb{R}^N \to \mathbb{R}^r$ be an arbitrary affine function

$$f(x) = Ax + b \quad (11)$$

with $A \in \mathbb{R}^{r \times N}$ and $b \in \mathbb{R}^r$. Then we have the following lemma [5], [13].

Lemma 1: The affine function (11) is uniquely determined by its values $f(v_i) = g_i, i = 1, \ldots, N+1$ at the vertices of S_N. Moreover, the restriction of f to S_N is a convex combination of its values at the vertices, and is given by

$$f(x) = GW^{-1} \begin{bmatrix} x \\ 1 \end{bmatrix}, \quad x \in S_N \quad (12)$$

where

$$G = [g_1 \quad \cdots \quad g_{N+1}] \quad (13)$$

$$W = \begin{bmatrix} v_1 & \cdots & v_{N+1} \\ 1 & \cdots & 1 \end{bmatrix} \quad (14)$$

are $r \times (N+1)$ and $(N+1) \times (N+1)$ real matrices.

Remark 3: Note that the restriction of an affine function f to a facet F_i of S_N (i.e., F_i itself is a simplex in $\mathbb{R}^{N-1}$) is affine, and for any $x \in F_i$, $f(x)$ is a convex combination of the values of f at the vertices of F_i.

Proposition 1: Let $w \in \mathbb{R}^r$ and $d \in \mathbb{R}$. Then $w^T f(x) > d$ everywhere in S_N, if and only if (iff) $w^T f(v_i) > d, i = 1, \ldots, N+1$.

Proof: The necessity follows immediately from the fact that the vertices $v_1, \ldots, v_{N+1}$ belong to S_N. For sufficiency, for any $x \in S_N$ we have

$$\begin{aligned} w^T f(x) &= w^T f\left(\sum_{i=1}^{N+1} \lambda_i v_i \right) \\ &= w^T \sum_{i=1}^{N+1} \lambda_i f(v_i) \sum_{i=1}^{N+1} \lambda_i w^T f(v_i) \\ &> d \sum_{i=1}^{N+1} \lambda_i = d. \end{aligned}$$

■

It is easy to see that the result of *Proposition 1* remains valid if $>$ is replaced by $\geq, =, <, \leq$. Also, it is obvious that *Proposition 1* remains valid if f is restricted to a facet F_i.

B. Affine Feedback-Control Laws in Simplices

In this section, we use the properties of affine functions presented above to completely describe the set of all affine vector fields with polyhedral bounds driving all initial states in a simplex through a desired facet in finite time, or making the simplex an invariant. We restrict our attention to affine functions (11) with $r = N$ defined on a simplex S_N and with values in a convex subset U of $\mathbb{R}^N$, i.e., to affine vector fields

$$\dot{x} = f(x), \quad f : S_N \to U \subseteq \mathbb{R}^N, \quad f(x) = Ax + b \quad (15)$$

where $A \in \mathbb{R}^{N \times N}$ and $b \in \mathbb{R}^N$. As stated before, if the vector field f is known at the vertices $(f(v_i) = g_i, i = 1, \ldots, N+1)$, then in (15), A is the matrix obtained by selecting the first N columns of GW^{-1}, while b is the last column of GW^{-1}, i.e.,

$$GW^{-1} = [A|b] \quad (16)$$

where G and W are given by (13) and (14), respectively.

Proposition 2 below gives a characterization of all affine vector fields driving all initial states in a simplex through a facet in finite time. Without restricting the generality of the problem, we assume that the exit facet is F_1.

Proposition 2 (Exit Through a Facet): There exists an affine vector field (15) driving all initial states in the simplex S_N through the facet F_1 in finite time, iff the convex sets $V_j^e, j = 1, \ldots, N+1$ are nonempty, where

$$V_1^e = U \bigcap \bar{V}_1^e \quad (17)$$

$$V_j^e = U \bigcap \bar{V}_j^e, \quad j = 2, \ldots, N+1 \quad (18)$$

with

$$\bar{V}_1^e = \{ g \in \mathbb{R}^N | n_j^T g \leq 0, j = 2, \ldots, \quad N+1, \text{ and } n_1^T g > 0 \} \quad (19)$$

$$\bar{V}_j^e = \{ g \in \mathbb{R}^N | n_k^T g \leq 0, k = 2, \ldots, \quad N+1, \quad k \neq j, \text{ and } n_1^T g > 0 \}. \quad (20)$$

Proof: The proof is given in the Appendix. ■

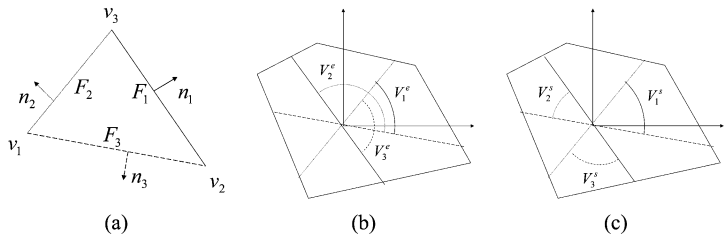

Fig. 2. When $N = 2$, the simplex defined by (10) is a triangle (a). For this example, the sets V_1, V_2, and V_3 from *Propositions 2* and *3* are the portions of cones shown in (b) and (c), respectively. V_1^e corresponds to the interior angle of the triangle at v_1, while V_2^e and V_3^e correspond to exterior angles at v_2 and v_3. V_1^s, V_2^s, and V_3^s correspond to interior angles at v_1, v_2, and v_3, respectively. The bounding polygon represents the (polyhedral) set U as in (15).

Remark 4: The conditions of *Proposition 2* guarantee that the trajectories of (15) leave the simplex S_N through F_1 the first time they hit F_1.

The following proposition characterizes all affine vector fields for which the simplex is an invariant.

Proposition 3 (Stay Inside a Simplex): There exists an affine vector field (15) on S_N whose trajectories never leave S_N, iff the convex sets $V_j^s, j = 1, \ldots, N+1$ are nonempty, where

$$V_j^s = U \bigcap \bar{V}_j^s, \quad j = 1, \ldots, N+1 \tag{21}$$

with

$$\bar{V}_j^s = \left\{ g \in \mathbb{R}^N | n_i^T g \leq 0, \quad i = 1, \ldots, N+1, i \neq j \right\}. \tag{22}$$

Proof: The proof is a simpler version of that given for *Proposition 2*, and it is omitted. ■

Remark 5: The nonemptiness conditions in *Propositions 2* and *3* are decoupled, i.e., V_k^e and V_k^s depend only on $g_k, k = 1, \ldots, N+1$. If one of the sets from *Propositions 2* and *3* is empty, then there is no affine vector field in S_N satisfying the corresponding property. If they are all nonempty, then any choice of $g_i \in V_i^{e,s}, i = 1, \ldots, N+1$ will give a valid [i.e., bounded, as in (15)] affine vector field by formula (12).

Intuitively, the conditions of *Proposition 2* state that the vector field at vertex v_1, which is opposite to exit facet F_1, should have a positive projection along the outer normal n_1 of facet F_1, and should point "inward" with respect to all the other facets $F_2, \ldots, F_{N+1}$. *Proposition 3* states that at all vertices, the vector field should point inward with respect to all facets containing the vertex. An illustrative description of the sets V_k^e and V_k^s from *Propositions 2* and *3* is given in Fig. 2 for the particular case of $N = 2$, when the simplex becomes a triangle.

Remark 6: If U is polyhedral, i.e., described by a set of linear inequalities, then the sets V_k^e and $V_k^s, k = 1, \ldots, N+1$ are all polyhedral, and checking their nonemptiness can be achieved using packages for polyhedra [18].

Proposition 4: 1) The sets $\bar{V}_i^e, i = 1, \ldots, N+1$ have a nonempty intersection with any open neighborhood of the origin in $\mathbb{R}^N$. 2) The intersection of any two sets $\bar{V}_i^s, i = 1, \ldots, N+1$ is the origin of $\mathbb{R}^N$.

Proof: The proof is given in the Appendix. ■

Proposition 5 (Constant Vector Fields): 1) There exists a constant vector field (15) satisfying the requirements of *Proposition 2*, iff V_1^e is nonempty. 2) There is no nonzero constant vector field (15) satisfying the requirements of *Proposition 3*.

Proof: There exists a constant vector field satisfying the requirements of *Propositions 2* or *3* iff $\bigcap_{i=1,\ldots,N+1} V_i^e \neq \emptyset$ or $\bigcap_{i=1,\ldots,N+1} V_i^s \neq \emptyset$, respectively. Indeed, $f(x) = g$, where g is an arbitrary element from the intersection, solves the problems. This being said, 1) follows immediately from the observation that $V_1^e \subseteq V_j^e$, for all $j = 2, \ldots, N+1$, and 2) is an obvious consequence of *Proposition 4* 2). ■

Therefore, as expected, there will never exist a nonzero constant vector field keeping system (15) inside the simplex for all times. See Fig. 2 for an illustration of these ideas for the particular case of $N = 2$, i.e., the simplices are triangles.

Proposition 6: 1) There exists a solution to *Proposition 2* for an arbitrary simplex iff U contains an open neighborhood of the origin in $\mathbb{R}^N$. 2) There exists a solution to *Proposition 3* for an arbitrary simplex iff U contains the origin in $\mathbb{R}^N$.

Proof: The sufficiency for 1) is immediate from *Proposition 4* 1). For the sufficiency of 2), if U contains the origin, then all sets V_i^s contain it, so the zero vector field solves *Proposition 3*. For necessity, assume by contradiction that U does not contain the origin, not even on the boundaries. Since U is convex, there exists a hyperplane, say H, passing through the origin, which leaves U on one side. Consider a simplex with facet F_1 contained in H and outer normal n_1 oriented on the opposite side of U. For such a simplex, all sets $V_k^e, k = 1, \ldots, N+1$ are empty, because they are all contained in $\{g \in \mathbb{R}^N | n_1^T g > 0\}$, which has an empty intersection with U. This contradicts that there is a solution to *Proposition 2*, and 1) is proved. If we now consider a simplex whose facet F_1 is contained in H with outer normal n_1 oriented toward the hyperspace containing U, then all the sets $V_k^s, k = 2, \ldots, N+1$ are empty, because they are all contained in $\{g \in \mathbb{R}^N | n_1^T g \leq 0\}$, which has an empty intersection with U. This contradicts that there is a solution to *Proposition 3*, and 2) is proved. ■

C. Construction of Triangular Affine Hybrid Systems

For the particular case of $N = 2$, *Proposition 6* leads to the following corollary, which is the main result of this paper.

Corollary 1: For an arbitrary triangulation of a polygon $\mathcal{P}$ (2), there exists an HS (4) with affine vector fields (9) producing the same language as the corresponding DG, i.e., $\mathcal{L}(\mathrm{HS}) = \mathcal{L}(\mathrm{DG})$, iff the convex set U giving the bounds of the vector fields contains an open neighborhood of the origin in $\mathbb{R}^2$.

Note that if the condition of *Corollary 1* is satisfied, then for each location $q_{ij} \in \mathcal{Q}$, there exists a whole set of vector fields $f_{q_{ii}}$ keeping the system in the triangle $I(l_i)$. Each choice of $g_k \in V_k^s$ given in *Proposition 3* will lead to a different vector field in

$I(l_i)$ according to (15). Similarly, for each location q_{ij} there exists a whole set of vector fields $f_{q_{ij}}$ driving the system from triangle $I(l_i)$ to its neighbor $I(l_j)$, and each choice of $g_k \in V_k^e$, as in *Proposition 2*, will lead to a different vector field in $I(l_i)$ according to (15).

In the next section, we present an algorithm for automatic generation of unique vector fields implementing an arbitrary string in the language $\mathcal{L}(\mathrm{DG})$.

IV. Algorithmic Generation of Vector Fields

In this section, we will use the extra degrees of freedom present in the characterization of the vector fields in *Corollary 1* to *guarantee smoothness of the produced trajectories*, if possible, and *minimize the time required for the accomplishment of a task* specified in terms of a fixed but arbitrary string $(l_{i_1}, l_{i_2}, \ldots, l_{i_m}) \in \mathcal{L}(\mathrm{DG})$.

To simplify the notation and without restricting the generality, assume that a fixed but arbitrary string in $\mathcal{L}(\mathrm{DG})$ is denoted by $(l_1, l_2, \ldots, l_m)$. To execute it, from the HS, we need to select the locations $q_{12}, q_{23}, \ldots, q_{(m-1)m}, q_{mm}$. Any of the corresponding vector fields $f_{q_{12}}, f_{q_{23}}, \ldots, f_{q_{(m-1)m}}, f_{q_{mm}}$ will definitely accomplish the task, as discussed in the previous section. However, even though the produced trajectories will be smooth inside each triangle, this property will, in general, be lost when transiting between adjacent triangles. Smoothness of trajectories is guaranteed everywhere in $\bigcup_{i=1}^{m} I(l_i)$ iff the vector fields $f_{q_{(i-1)i}}, f_{q_{i(i+1)}}$ match on the separating facet $I(l_{i-1}) \bigcap I(l_i)$ for all $i = 2, \ldots, m-1$, and the vector fields $f_{q_{(m-1)m}}, f_{q_{mm}}$ match on $I(l_{m-1}) \bigcap I(l_m)$. This guarantees the continuity of the vector field everywhere in $\bigcup_{i=1}^{m} I(l_i)$, and therefore, the produced trajectories are C^1 (differentiable with continuous derivatives), or smooth. Using *Lemma 1*, and noting that the separating facets are S_1 triangles (or line segments), the matching condition everywhere on a separating facet is satisfied if and only it is satisfied at the vertices. This implies that matching can be achieved for a whole sequence iff all the polyhedral sets obtained as solutions of *Propositions 2* or *3* for a given point, which can be a vertex of several triangles, have a nonempty intersection.

In what follows, we present an algorithm that takes as input a set of points and a relation assigning these points to a sequence of pairwise adjacent triangles, and outputs a set of vector fields guaranteeing smoothness of the corresponding trajectories in as large as possible subsequences of triangles. Let $p_1, \ldots, p_{m+2} \in \mathbb{R}^2$ denote the coordinates of the vertices of triangles $I(l_1), \ldots, I(l_m)$ in a reference frame $\{F\}$. Let $\mathcal{A} \subset \{1, \ldots, m+2\} \times \{1, \ldots, m\} \times \{1, 2, 3\}$ be a relation describing the assignment of the points $p_1, \ldots, p_{m+2} \in \mathbb{R}^2$ as vertices of the triangles $I(l_1), \ldots, I(l_m)$ with the following significance: $(i, j, k) \in \mathcal{A}$ means that p_i is a vertex of triangle $I(l_j)$ with rank k, which we denote by v_k^j. The rank $k, k = 1, 2, 3$ of a vertex v_k^j of triangle $I(l_j)$ is defined as follows. The vertex of rank 1 (v_1^j) of triangle $I(l_j), j = 1, \ldots, m-1$ is not a vertex of $I(l_{j+1})$. For $I(l_m)$, the vertex of rank 1 (v_1^m) does not belong to triangle $I(l_{m-1})$. Ranks 2 and 3 (v_2^j and v_3^j) are defined so that if $(i_1, j, 1), (i_2, j, 2), (i_3, j, 3) \in \mathcal{A}$, then p_{i_1}, p_{i_2}, and p_{i_3} are coordinates of vertices of triangle $I(l_j)$ in counterclockwise order. See Fig. 3(a) and (c) for two examples of point-vertex assignment using the notation described above. Corresponding to this assignment of vertices, for each triangle $I(l_j), j = 1, \ldots, m$, we define three facets F_k^j with outer normals n_k^j, where F_k^j is the facet opposite to vertex v_k^j of triangle $I(l_j), k = 1, 2, 3$.

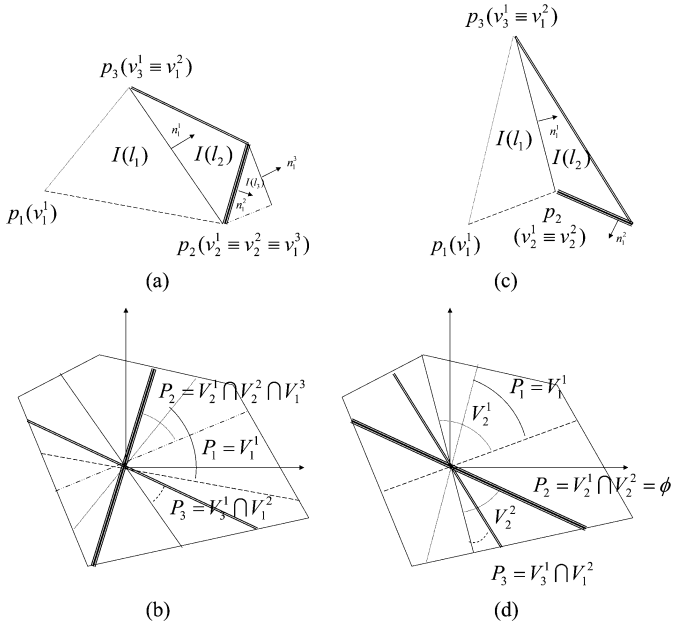

Fig. 3. Examples of adjacent triangle sequences. (a) and (b) show an example where the matching condition can be satisfied. (c) and (d) illustrate a situation when matching is not possible.

In what follows, we assume that the set U is polyhedral. This will allow for the development of a computationally efficient algorithm based only on polyhedral operations. In general, if U is not polyhedral, one can start with a polyhedral underapproximation. Let $P_i, i = 1, \ldots, m+2$ denote the polyhedral set for point p_i, and V_k^j the polyhedral set obtained by applying *Propositions 2* or *3* to the vertex v_k^j of triangle $I(l_j)$. In Table I, we present an algorithm that takes as input the set of coordinates $\{p_1, \ldots, p_{m+2}\}$ and the triangle-vertex relation $\mathcal{A}$, and returns the maximal subsequence $\{1, \ldots, j_1\}$ of triangle indexes for which matching conditions can be satisfied, i.e., smooth trajectories can be achieved. The main idea is that the triangles are visited in the given order, starting from $j = 1$, and restrictions V_k^j are added to sets P_i corresponding to the points p_i, which act as vertices v_k^j corresponding to *Propositions 2* or *3*. When, in a given triangle j, the set P_i corresponding to a point becomes empty, then we stop, set $j_1 = j$, and keep the nonempty sets P_i from the previous step, which guarantees that smooth trajectories bring all initial conditions from triangle $I(l_1)$ to $I(l_{j_1})$ in finite time. Then the algorithm can be reiterated, starting from j_1, to produce another subsequence, and finally provide a solution to *Problem 1* with a minimum number of subsequences. Of course, at the facet separating $I(l_{j_1})$ and $I(l_{j_1+1})$, the vector field will be discontinuous.

There are two important points we need to make with regard to the matching condition. First, as stated in *Proposition 5*, if *Proposition 2* is used in just one triangle, and the constraint set U is such that the sets V_1^e, V_2^e, and V_3^e are nonempty, then it is

TABLE I
ALGORITHM FOR DETERMINING A MAXIMAL SEQUENCE OF TRIANGLES FOR WHICH SMOOTH TRAJECTORIES CAN BE GENERATED

Determine maximal smooth subsequence $(p_1, p_2, \ldots, p_{m+2}, \mathcal{A})$

$P_i \leftarrow \mathbb{R}^2$, for all $i = 1, \ldots, m+2$ (* *initialize polyhedral sets for all points* *)
$j \leftarrow 1$ (* *start with first triangle* *)
while P_i is nonempty, for all $i = 1, \ldots, m+2$
  $v_k^j \leftarrow p_i$, for all $(i, j, k) \in \mathcal{A}$ (* *identify the vertices of triangle* j *)
  if $j = m$ (* *last triangle* *)
    apply Proposition 3 in triangle j with vertices v_k^j to obtain V_k^j, $k = 1, 2, 3$
  else (* *not last triangle* *)
    apply Proposition 2 in triangle j with vertices v_k^j to obtain V_k^j, $k = 1, 2, 3$
  endif
  $P_i \leftarrow P_i \bigcap V_k^j$, for all $(i, j, k) \in \mathcal{A}$ (* *update allowed polyhedral sets for* p_i*'s which are vertices of triangle* j *)
  $j \leftarrow j + 1$ (* *move to the next triangle* *)
endwhile
$S \leftarrow \{1, 2, \ldots, j-1\}$ (* *sequence of triangles for which smooth trajectories can be designed* *)
(* P_i, $(i, j, k) \in \mathcal{A}$, $j \in S$ *are nonempty polyhedral sets for the points which are vertices of the triangles in sequence* S *)*

Construction of continuous vector fields (S)

for all i so that $(i, j, k) \in \mathcal{A}$ and $j \in S$ **do**
  $c_i \leftarrow \sum_{j \in S, (i,j,k) \in \mathcal{A}} n_1^j$ (* *sum of all outer normals to exit facets to which* p_i *is a vertex* *)
  the velocity g_i at p_i is the solution to the following LP: $\max_{g_i} c_i^T g_i$, $g_i \in P_i$
endfor
for all $j \in S$ **do**
  using (12) construct $f_{q_{j(j+1)}}(x)$ so that $f_{q_{j(j+1)}}(v_k^j) = g_i$, $(i, j, k) \in \mathcal{A}$
endfor

always possible to construct a constant vector field solving the problem, based on the fact that $V_1^e \subset V_2^e$ and $V_1 \subset V_3^e$ always. However, if matching is desired with subsequent triangles in a sequence, then the inclusion above might not be valid anymore, and affine feedback controllers with explicit state dependence are necessary. A graphical illustration of this idea is given in Fig. 3(a) and (b), where point p_3 is a vertex of rank 3 in $I(l_1)$, and of rank 1 in $I(l_2)$. If just the problem of reaching facet F_1^1 of $I(l_1)$ was considered, then $V_1^1 \subset V_3^1$. However, if matching is required for the sequence $I(l_1), I(l_2), I(l_3)$, then the allowed set of p_3 is $P_3 = V_3^1 \bigcap V_1^2$, which has an empty intersection with P_1. Therefore, the affine vector field $f_{q_{12}}$ in $I(l_1)$ cannot be chosen constant anymore.

Second, for the particular case of triangles in plane that we consider in this paper, there is a simple geometrical interpretation of the matching condition. It is violated iff there exists a sequence of adjacent triangles which "rotates" around a common vertex with more than π. See Fig. 3(c) and (d) for a graphical illustration of such a situation.

To minimize the time spent on the produced trajectories, from the polyhedral sets P_i corresponding to each point p_i in a subsequence where the matching condition is satisfied, we select a velocity vector which has a maximum projection along a weighted sum of all outward normals of all exit facets of which the point is a vertex. This problem is a linear program (LP), and has a unique solution. A lower bound for the projection of velocity at vertices along a constant vector is a lower bound for the projection of the affine vector field everywhere in the triangle, by the convexity property of *Lemma 1*. The algorithm shown in Table I also returns the corresponding vector fields which guarantee smoothness of trajectories in the subsequence and maximization of speed.

Remark 7 (Computational Issues): The algorithmic framework for generation of provably correct vector fields in a polygonal environment involves two steps: *triangulation* and *generation of vector fields*. The triangulation procedure is computationally inexpensive. A simply connected polygon can be triangulated in $O(n)$ time, where n denotes the number of vertices [7]. The running time of the best known algorithm for triangulating a polygon with h holes is $O(n + h(\log h)^{1+\epsilon})$ [4]. In the case when U is polyhedral, checking the existence of a HS as in *Definition 3*, reduces to $|\mathcal{Q}| = |L| + |t|$ LP feasibility checks. If smoothness and time spent are not of interest, then any choice of elements from these sets produce vector fields for the HS using (15). If smoothness and maximization of speed is desired, then the vector fields are constructed according to the algorithms described in Table I. For a sequence of adjacent triangles in which smooth trajectories can be generated, we solve a number of LPs equal to the number of polygon vertices pertaining to the triangles. The number of linear constraints in each of these LPs varies, and depends on how many triangles in the sequence share the corresponding vertex.

V. ROBOT CONTROL

For the fully actuated robot (1), the feedback controllers solving *Problem 1* are given by the vector fields of the HS constructed as shown in the previous sections. From *Corollary 1*, we have the following.

Corollary 2: For a fully actuated robot (1) with control bounds modeled as a convex set U, there exists a solution to *Problem 1* for arbitrary polygons and triangulations if the set U contains an open neighborhood of the origin in $\mathbb{R}^2$.

Note that the necessary and sufficient condition in *Corollary 1* becomes sufficient in the above corollary, since *Corollary 1* only holds for the class of affine feedback-control systems. In other words, the above corollary transforms into an equivalent condition if the feedback-control laws are restricted to the class

of affine systems. Also, the condition of *Corollary 2* is in accordance with one's intuition: if the robot is able to move in all directions, then it can execute arbitrary strings. What is not at all intuitive, is the fact that it can do this under *affine feedback control*, and that *convex control bounds* can be guaranteed. Moreover, the robot can execute certain strings under affine feedback even if the above condition is not satisfied. The equivalent conditions and analytical formulas for automatic generation of feedback-control laws were presented in the previous sections.

Let us now consider a differentially driven wheeled robot (unicycle) described by $(R, d) \in \mathrm{SE}(2)$ in the world frame, where $d \in \mathbb{R}^2$ gives the position vector of the robot center and $R \in \mathrm{SO}(2)$ is the rotation of the robot frame. The control $w = [w_1, w_2]^T \subset W \subseteq \mathbb{R}^2$ consists of driving (w_1) and steering (w_2) speeds, where W is a set capturing control bounds.

It is well known that this underactuated system with state (R, d) and control w is uncontrollable. For this reason, as in [10], we define a reference point different from the robot center, and with coordinates $(\epsilon, 0)(\epsilon > 0)$ in the robot frame and $x = [x_1, x_2]^T$ in the world frame. It can be shown that the velocity $u(x)$ of the reference point is related to the initial controls w by a nonsingular map

$$w = E^{-1} R^T u(x) \tag{23}$$

where $E = [1, 0; 0, \epsilon]$.

As in the fully actuated case, $u(x)$ is determined by the construction of the HS (4), and particular strings can be implemented as shown in Section IV. Using (23), it is easy to see that bounds U on the velocity of the reference point u easily translate to bounds W on the original control w, by noting that they are related by a rotation and a diagonal scaling matrix (dependent on ϵ) with positive diagonal entries. The origin is mapped to the origin through both these maps, and therefore, by applying *Corollary 2* to the reference point, we have the following.

Corollary 3: For a unicycle with control bounds W, there exists a solution to *Problem 1* for arbitrary polygons and triangulations, if the set W contains an open neighborhood of the origin in $\mathbb{R}^2$.

Again, the intuition works here as well. A unicycle can execute arbitrary strings over the DG induced by a triangulation of its polygonal observable space if it can rotate both left and right and translate both forward and backward.

VI. Simulation Results

Consider a unicycle with driving and steering speeds w_1 and w_2 limited to 1 and 2, respectively. In other words, $W = [-1, 1] \times [-2, 2]$. Assume that the displacement of the reference point in the unicycle frame is $\epsilon = 0.5$. Then, it is easy to see that with a bit of conservatism, the rectangular bounds $U = [-\sqrt{2}/2, \sqrt{2}/2]^2$ for the reference point will guarantee the imposed control bounds W. Indeed, under all planar rotations R, U becomes a disk centered at 0 with radius 1, which is then scaled to an ellipse with semi-axes 1 and 2, according to (23). The actual controls of the robot are inside an ellipse centered at 0 with semi-axes 1 and 2, which is contained in the rectangle W. Therefore, the initial control bounds are guaranteed if U is chosen as above.

A. Simple Environment

To illustrate the assignment of vector fields and the satisfaction of matching conditions and control bounds, we first consider a simple polygonal environment consisting of the sequence of adjacent triangles shown at the top of Fig. 4. This example can be also be interpreted as the execution of a string from the language of a DG of a larger triangulated polygon. Δ_1 denotes the initial triangle, and Δ_{14} is the final triangle.

By applying the algorithm given in Table I, we determined that the maximal smooth sequence starting at Δ_1 is $\Delta_1, \Delta_2, \ldots, \Delta_9$, with stop in Δ_9. Indeed, it is easy to see that if exit through the common facet of Δ_9 and Δ_{10} was desired, then the rotation around the common vertex of $\Delta_7, \Delta_8, \Delta_9$, and Δ_{10} would be larger than π. The produced vector fields guaranteeing smooth motion in the sequence $\Delta_1, \Delta_2, \ldots, \Delta_9$ are plotted in Fig. 4, middle left. Then the algorithm is reiterated, and the vector fields corresponding to the next smooth sequence $\Delta_9, \Delta_{10}, \ldots, \Delta_{14}$, with stop in Δ_{14}, are shown in Fig. 4, middle right. Note that the vector fields on adjacent triangles match on the separating facet in each of the subsequences shown in Fig. 4 in the middle row.

The motion of a unicycle arbitrarily initialized in Δ_1 is shown in Fig. 4, top. The corresponding velocity u of the reference point x and the controls w are shown in Fig. 4, bottom left and right, respectively. It is easy to see that each component of u and w are continuous everywhere, except for a time close to 2500, when the vector field in Δ_9 is switched from a stopping one, as in Fig. 4, middle left, to a driving one, as in Fig. 4, middle right. Also, note that the polyhedral bounds for u and w are satisfied for all times during the produced motion.

B. Complex Environment

To illustrate the computational efficiency of the developed algorithms and the usefulness of the created framework, we consider a more realistic example, such as the one shown in Fig. 5 (left). The outer polygonal line represents the boundaries of the environment, while the inner closed polygonal lines model obstacles. The obtained polygon, which has 44 vertices, was triangulated using the algorithm available in [27]. The resulting triangulation, which consists of 46 triangles, and the corresponding DG are shown in Fig. 5 (left). Sample trajectories of the unicycle described at the beginning of this section, implementing strings in the language of this DG, are shown in Fig. 5 (right).

VII. Conclusion

In this paper, we proposed a method for algorithmically generating and verifiably composing affine feedback-control laws for planar robots operating in polygonal environments. In addition to being computationally efficient, our solution formally relates the high-level plans and low-level motions using modern tools from HS theory. Future work includes extensions on the discrete side, resulting in more complicated plans, such as temporal logic planning [32] and games on graphs [17]. On the continuous side, we will extend this framework toward more complicated systems, such as underactuated robots and second-order dynamics.

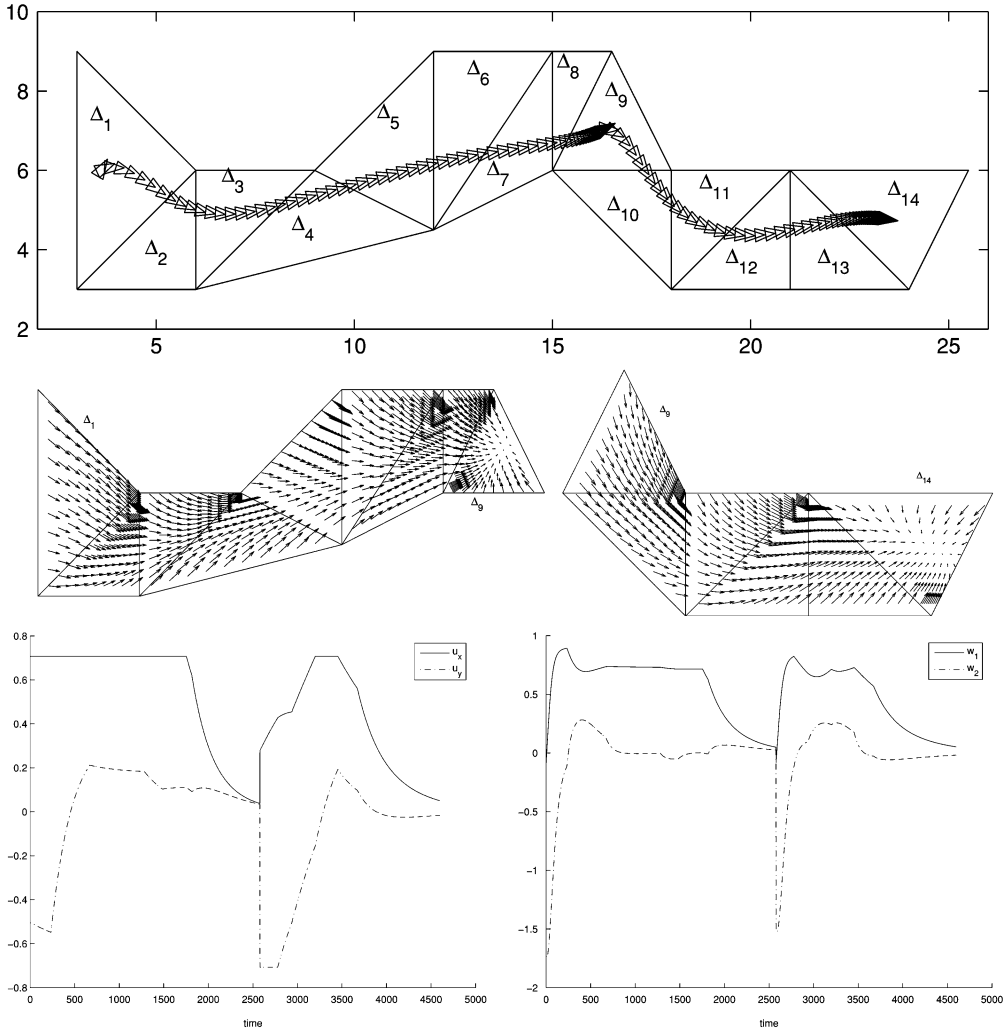

Fig. 4. Top: Sequence of adjacent triangles and an example of unicycle motion. Middle: Vector fields obtained by applying the algorithms in Table I to this sequence of triangles. Left: smooth sequence $\Delta_1, \Delta_2, \ldots, \Delta_9$, with stop in Δ_9; Right: smooth sequence $\Delta_9, \Delta_{10}, \ldots, \Delta_{14}$, with stop in Δ_{14}. Bottom: Left: Time evolution of reference point velocity u; Right: Time evolution of driving and steering controls w_1 and w_2.

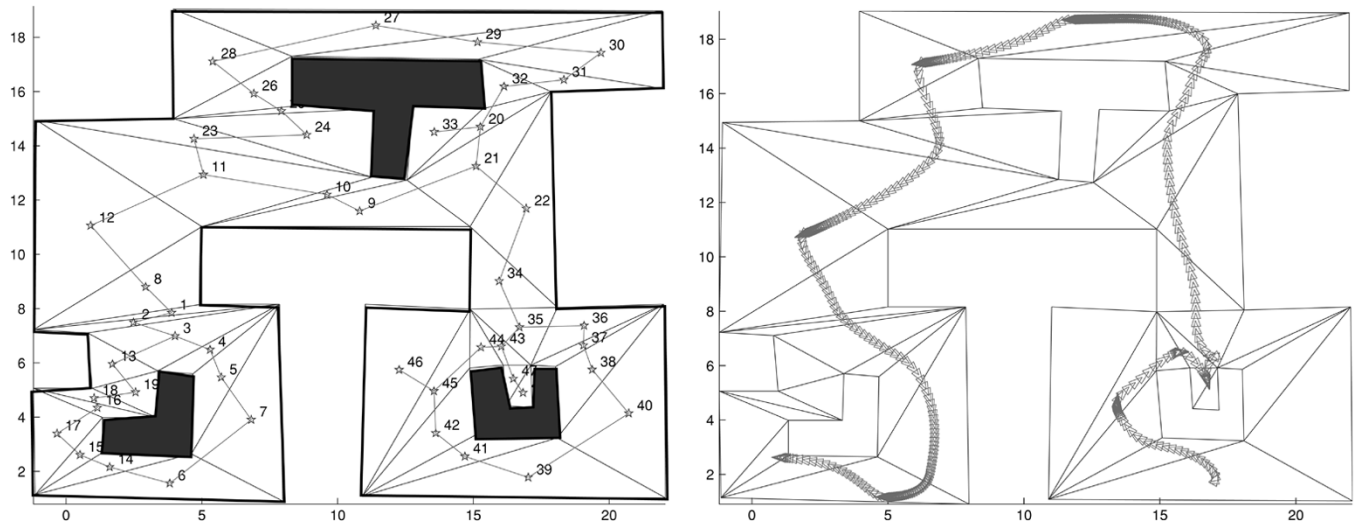

Fig. 5. Left: Polygonal environment, its triangulation, and the dual of the triangulation. Right: Sample trajectories.

APPENDIX

The following lemma states a well-known result [25] and is used in the proofs given below.

Lemma 2: In any simplex S_N, for an arbitrary $i = 1,\dots,N+1$, the vectors $n_j, j = 1,\dots,N+1, j \neq i$ are linearly independent. Moreover, n_i is a strictly negative linear combination of $n_j, j = 1,\dots,N+1, j \neq i$.

Proof of Proposition 2: A related proof of this result can be found in [5] and [13]. First, note that V_i^e are convex, since V is convex and $\bar{V}_i^e$ are (convex) polyhedra, for all $i = 1,\dots,N+1$. For sufficiency, if the sets V_i^e are all nonempty, then choose arbitrary $g_i \in V_i^e, i = 1,\dots,N+1$ and construct the unique affine function (12) in S_N satisfying $f(v_i) = g_i, i = 1,\dots,N+1$. Since for every $x \in S_N, f(x)$ is a convex combination of $g_1,\dots,g_{N+1} \in V, f(x)$ is contained in the convex hull of $g_1,\dots,g_{N+1}$. This is the smallest convex set containing $g_1,\dots,g_{N+1}$, and therefore is included in V. So, $f(x) \in V, \forall x \in S_N$, as required. The restriction of $f(x)$ to an arbitrary facet $F_k, k = 2,\dots,N+1$ is, of course, an affine function, therefore, a convex combination of its values g_j at the corresponding vertices $v_j, j = 1,\dots,N+1, j \neq k$. Since $n_k^T g_j \leq 0, k = 2,\dots,N+1, j \neq k$, using *Proposition 1*, we conclude that $n_k^T f(x) \leq 0$ everywhere on F_k, so they cannot leave through the facet $F_k, k = 2,\dots,N+1$. On the other hand, since $n_1^T g_j > 0, j = 1,\dots,N+1$, we conclude that $n_1^T f(x) > 0, \forall x \in S_N$. Therefore, all trajectories of (15) will have a positive speed of motion toward F_1 everywhere in S_N, which implies that the simplex will eventually be left.

For necessity, assume there is an affine vector field (15) driving all states in S_N through F_1 in finite time. Let $f(v_i) = g_i, i = 1,\dots,N+1$. Of course $g_i \in V$, since $f(x) \in V$ everywhere in S_N by hypothesis. We will show that g_i satisfies the inequalities of $\bar{V}_i^e, i = 1,\dots,N+1$, so all sets V_i^e are nonempty. If we assume that there exists $j = 2,\dots,N+1$ so that $n_j^T g_1 > 0$, then (15) initialized at v_1 (or very close to v_1 on F_j) will leave the simplex without hitting F_1 (by continuity). Therefore, $n_j^T g_1 \leq 0, \forall j = 2,\dots,N+1$. Similarly, for an arbitrary $j = 2,\dots,N+1, n_k^T g_j \leq 0, \forall k = 2,\dots,N+1, k \neq j$ because otherwise, there will exist points close to v_j on F_k leaving the simplex. It is obvious that we need to have $n_1^T f(x) > 0$ everywhere on the exit facet F_1, which implies $n_1^T g_j > 0, \forall j = 2,\dots,N+1$. The only thing that remains to be proved is $n_1^T g_1 > 0$. Assume by contradiction that $n_1^T g_1 \leq 0$. According to *Lemma 2*, n_1 is a negative linear combination of $n_2,\dots,n_{n+1}$ and we can write $n_1 = \sum_{i=2}^{N+1} \mu_i n_i$, where $\mu_i < 0, i = 2,\dots,N+1$. This leads to $\sum_{i=1}^{N+1} \mu_i n_i^T g_1 \leq 0$. However, we have already proved that $n_i^T g_1 \leq 0$, for all $i = 2,\dots,N+1$, from which we conclude that $\mu_i n_i^T g_1 = 0$, for all $i = 2,\dots,N+1$. Since $n_2,\dots,n_{n+1}$ are linearly independent, it follows that $g_1 = 0$, i.e., the vector field at the vertex v_1 is zero. This means that the system initialized at v_1 will stay there forever, and therefore, will not leave the simplex in finite time, which contradicts the hypothesis, and the proposition is proved. ∎

Proof of Proposition 4: It is easy to see that in *Proposition 2*, $\bar{V}_1^e \subseteq \bar{V}_j^e$ (which also implies $V_1^e \subseteq V_j^e$), for all $j = 2,\dots,N+1$. Therefore, it is enough to prove 1) for $\bar{V}_1^e$. Let

$$C = \left\{ g \in \mathbb{R}^N | n_j^T g \leq 0, j = 2,\dots,N+1 \right\}.$$

It is easy to see that C is a cone with apex 0. Also

$$\bar{V}_1^e = C \setminus \{0\} \tag{24}$$

i.e., $\bar{V}_1^e$ is the cone C from which the apex has been removed. Indeed, any $g \in \bar{V}_1^e$ satisfies $g \in C \setminus \{0\}$, since $n_1^T g > 0$ guarantees $g \neq 0$. Therefore, $\bar{V}_1^e \subseteq C \setminus \{0\}$. For an arbitrary $g \in C \setminus \{0\}$, by *Lemma 2*, $n_1^T g = \sum_{i=2}^{N+1} \mu_i n_i^T g$, where $\mu_i < 0, i = 2,\dots,N+1$. Each term in this sum is larger or equal to zero. The sum can, therefore, be equal to zero iff each term is zero, which implies $n_i^T g = 0$, for all $i = 1,\dots,N+1$. This can only happen if $g = 0$, since $n_i, i = 2,\dots,N+1$ are linearly independent by *Lemma 2*. But $g \neq 0$, therefore $C \setminus \{0\} \subseteq \bar{V}_1^e$. (24) is proved, which immediately implies 1).

For 2), let $i, j \in \{1,\dots,N+1\}, i \neq j$. If $g \in \bar{V}_i^s \bigcap \bar{V}_j^s$, then $n_k^T g \leq 0$ for all $k = 1,\dots,N+1$. Since, by *Lemma 2*, n_1 is a negative linear combination of $n_2,\dots,n_{N+1}$, it follows that $n_1^T g = \sum_{i=2}^{N+1} \mu_i n_i^T g$, with $\mu_i < 0, i = 2,\dots,N+1$. The left-hand side of this equality is ≤ 0, while the right-hand side is ≥ 0, and since $n_2,\dots,n_{N+1}$ are linearly independent, it follows that $g = 0$, and 2) is proved. ∎

REFERENCES

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Oliviero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoret. Comput. Sci.*, vol. 138, pp. 3–34, 1995.

[2] R. Alur, T. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proc. IEEE*, vol. 88, no. 2, pp. 971–984, Jul. 2000.

[3] A. Astolfi, "Discontinuous control of nonholonomic systems," *IEEE Trans. Autom. Control*, vol. 27, no. 1, pp. 37–45, Jan. 1996.

[4] R. Bar-Yehuda and B. Chazelle, "Triangulating disjoint Jordan chains," *Int. J. Computat. Geom. Appl.*, vol. 4, no. 4, pp. 475–481, 1994.

[5] C. Belta and L. C. G. J. M. Habets, "Constructing decidable hybrid systems with velocity bounds," in *Proc. 43rd IEEE Conf. Decision Control*, Nassau, Bahamas, 2004.

[6] R. Burridge, A. Rizzi, and D. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *Int. J. Robot. Res.*, vol. 18, no. 6, pp. 534–555, Jun. 1999.

[7] B. Chazelle, "Triangulating a simple polygon in linear time," *Disc. Comput. Geom.*, vol. 6, pp. 485–524, 1991.

[8] P. Cheng, Z. Shen, and S. M. LaValle, "RRT-based trajectory design for autonomous automobiles and spacecraft," *Arch. Control Sci.*, vol. 11(XLVII), no. 3–4, pp. 167–194, 2001.

[9] D. C. Conner, A. A. Rizzi, and H. Choset, "Composition of local potential functions for global robot control and navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 3, Las Vegas, NV, 2003, pp. 3546–3551.

[10] J. Desai, J. P. Ostrowski, and V. Kumar, "Controlling formations of multiple mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 4, Leuven, Belgium, 1998, pp. 2864–2869.

[11] J. A. Fernandez and J. Gonzalez, "Multihierarchical graph search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 103–113, Jan. 2002.

[12] C. Galindo, J. A. Fernandez, and J. Gonzalez, "Hierarchical task planning through world abstractions," *IEEE Trans. Robot.*, vol. 20, no. 4, pp. 667–690, Aug. 2004.

[13] L. C. G. J. M. Habets and J. H. van Schuppen, "A control problem for affine dynamical systems on a full-dimensional polytope," *Automatica*, vol. 40, pp. 21–35, 2004.

[14] E. Haghverdi, P. Tabuada, and G. Pappas, "Bisimulation relations for dynamical and control systems," in *Electronic Notes in Theoretical Computer Science*. New York: Elsevier, 2003, vol. 69.

[15] V. Isler, C. Belta, K. Daniilidis, and G. J. Pappas, "Stochastic hybrid control for visibility-based pursuit-evasion games," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 2, Sendai, Japan, 2004, pp. 1432–1437.

[16] V. Isler, S. Kannan, and S. Khanna, "Locating and capturing an evader in a polygonal environment," in *Proc. Workshop Algorithmic Found. Robot.*, 2004, pp. 351–367.
[17] ——, "Randomized pursuit-evasion with limited visibility," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, 2004, pp. 1053–1063.
[18] B. Jeannet, "Convex Polyhedra Library," Verimag, Grenoble, France, Tech. Rep., 1999.
[19] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, pp. 90–98, 1986.
[20] D. E. Koditschek, "The control of natural motion in mechanical systems," *ASME J. Dyn. Syst., Meas., Control*, vol. 113, no. 4, pp. 548–551, 1991.
[21] G. A. Lafferriere and H. Sussmann, "A differential geometric approach to motion planning," in *Nonholonomic Motion Planning*, Z. Li and J. F. Canny, Eds. Norwell, MA: Kluwer, 1993, pp. 235–270.
[22] F. Lamiraux and J. P. Laumond, "Smooth motion planning for car-like vehicles," *IEEE Trans. Robot. Autom.*, vol. 17, no. 4, pp. 498–502, Aug. 2001.
[23] J. C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.
[24] S. M. LaValle and M. S. Branicky, "On the relationship between classical grid search and probabilistic roadmaps," *Int. J. Robot. Res.*, vol. 23, no. 78, pp. 673–692, 2004.
[25] M. Overmars, M. de Berg, M. van Kreveld, and O. Schwarzkopf, *Computational Geometry, Algorithms and Applications*. New York: Springer-Verlag, 1997.
[26] R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Trans. Autom. Control*, vol. 38, no. 5, pp. 700–716, May 1993.
[27] A. Narkhede and D. Manocha. Fast polygon triangulation based on Seidel's algorithm. [Online]. Available: ftp://ftp.cs.unc.edu/pub/users /manocha/CODE/Triangulation/
[28] J. O'Rourke, *Computational Geometry in C*. Cambridge, U.K.: Cambridge Univ. Press, 1998.
[29] G. J. Pappas, "Bisimilar linear systems," *Automatica*, vol. 39, no. 12, pp. 2035–2047, 2003.
[30] P. Rouchon, M. Fliess, J. Levine, and P. Martin, "Flatness, motion planning and trailer systems," in *Proc. 32nd IEEE Conf. Decision Control*, Austin, TX, Dec. 1993, pp. 2700–2705.
[31] L. D. Seneviratne, W.-S. Ko, and S. W. Earles, "Triangulation-based path planning for a mobile robot," *J. Mech. Eng. Sci.*, pt. C, vol. 211, no. 5, pp. 365–371, 1997.
[32] P. Tabuada and G. Pappas, "Model checking LTL over controllable linear systems is decidable," in *Lecture Notes in Computer Science*. New York: Springer-Verlag, 2003, vol. 2623.
[33] J. Thomas, A. Blair, and N. Barnes, "Toward an efficient optimal trajectory planner for multiple mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 3, 2003, pp. 2291–2296.
[34] D. Tilbury and A. Chelouah, "Steering a three input nonholonomic system using multirate controls," in *Proc. Eur. Control Conf.*, Groningen, The Netherlands, 1992, pp. 1993–1998.

Calin Belta received the B.S. and M.Sc. degrees in control and computer science from the Technical University of Iasi, Iasi, Romania, the M.Sc. degree in electrical engineering from Louisiana State University, Baton Rouge, and the M.Sc. and Ph.D. degrees in mechanical engineering from the University of Pennsylvania, Philadelphia.

He is currently an Assistant Professor in the Department of Manufacturing Engineering, Boston University, Boston, MA. His research interests include planning and control for formations of robots, hybrid systems, and biomolecular networks.

Dr. Belta received a National Science Foundation CAREER award in 2005, a Fulbright study award in 1997, and was the Valedictorian of his class in 1995. He received the Best Paper Award at the International Conference on Systems Biology in 2004, and was a finalist for the ASME Design Engineering Technical Conference Best Paper Award in 2002, and for the Anton Philips Best Student Paper Award at the IEEE International Conference on Robotics and Automation in 2001.

Volkan Isler (M'01) received the M.S.E. and Ph.D. degrees in computer and information science from the University of Pennsylvania, Philadelphia, and the B.S. degree in computer engineering from Bogazici University, Istanbul, Turkey.

He is currently a Postdoctoral Researcher with the Center for Information Technology Research in the Interest of Society (CITRIS), University of California, Berkeley. His research interests are in robotics (pursuit evasion, exploration, motion planning), sensor networks (deployment, target tracking, and localization) and computer vision (tele-immersion, model reconstruction, and segmentation).

George J. Pappas (S'91-M'98–SM'04) received the Ph.D. degree from the University of California at Berkeley in December 1998.

In 1999, he was a Postdoctoral Researcher at the University of California at Berkeley and the University of Pennsylvania, Philadelphia. In 2000, he joined the University of Pennsylvania as an Assistant Professor in the Department of Electrical and Systems Engineering, where he is currently an Associate Professor and the Graduate Group Chair. He also holds a secondary appointments in the Departments of Computer and Information Sciences, and Mechanical Engineering and Applied Mechanics. He has published over 100 articles in the areas of hybrid systems, hierarchical control systems, distributed control systems, nonlinear control systems, and geometric control theory, with applications to flight management systems, robotics, and unmanned aerial vehicles. He co-edited *Hybrid Systems: Computation and Control* (New York: Springer-Verlag, 2004, ser. Lecture Notes in Computer Science).

Dr. Pappas is the recipient of a National Science Foundation (NSF) Career Award in 2002, as well as the 2002 NSF Presidential Early Career Award for Scientists and Engineers (PECASE). He received the 1999 Eliahu Jury Award for Excellence in Systems Research from the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley. His and his student's papers were finalists for the Best Student Paper Award at the 1998 IEEE Conference on Decision and Control, the 2001 American Control Conference, the 2001 IEEE Conference on Decision and Control, the 2004 American Control Conference, and the 2004 IEEE Conference on Decision and Control. He is currently an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATIC CONTROL.